

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problems Mailbox.**



PATENT ABSTRACTS OF JAPAN

(11) Publication number: **08087271 A**(43) Date of publication of application: **02 . 04 . 96**

(51) Int. Cl.

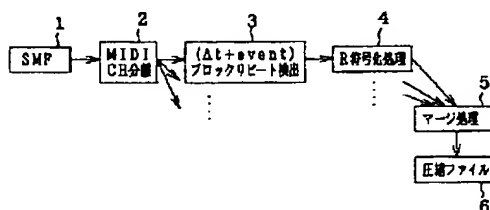
G10H 1/00**G10K 15/04****H03M 7/46**(21) Application number: **06248875**(22) Date of filing: **17 . 09 . 94**(71) Applicant: **VICTOR CO OF JAPAN LTD**(72) Inventor: **HIKAWA KAZUO
KUROIWA TOSHIO**(54) **COMPRESSION METHOD AND EXPANSION
METHOD FOR PLAYING INFORMATION**

COPYRIGHT: (C)1996,JPO

(57) Abstract:

PURPOSE: To compress playing information in order to reduce the data amount of playing information which specifies the interval of sound constituting musical tone and the sounding time by deleting a repeated part so as not to overlap the repetition part when the existence of a repetition is detected.

CONSTITUTION: SMFs of many musics are stored in a memory. An SMF1 equivalent to one music specified by a control system is separated every channel by an MIDI CH separation block 2. A block 3 detects a ($\Delta t + event$) block repeat and when there exists a repetition in the playing information pattern, the block 3 detects the repetition. A block 4 performs an R(repeat) coding processing, deletes the repeated part detected by the block 3 so as not to overlap the repetition part and adds necessary signals. A merge processing block 5 merges the signals to which multi-channel R coding processings are performed into an original SMF format. Thus, Δt is integrated into one body and a compressed file 6 is produced.



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平8-87271

(43) 公開日 平成8年(1996)4月2日

(51) Int.Cl. ⁹	識別記号	庁内整理番号	F I	技術表示箇所
G 1 0 H 1/00	1 0 2 Z			
G 1 0 K 15/04	3 0 2 D			
H 0 3 M 7/46		9382-5K		

審査請求 未請求 請求項の数7 F D (全 16 頁)

(21) 出願番号 特願平6-248875

(22) 出願日 平成6年(1994)9月17日

(71) 出願人 000004329

日本ビクター株式会社

神奈川県横浜市神奈川区守屋町3丁目12番地

(72) 発明者 飛河 和生

神奈川県横浜市神奈川区守屋町3丁目12番地 日本ビクター株式会社内

(72) 発明者 黒岩 俊夫

神奈川県横浜市神奈川区守屋町3丁目12番地 日本ビクター株式会社内

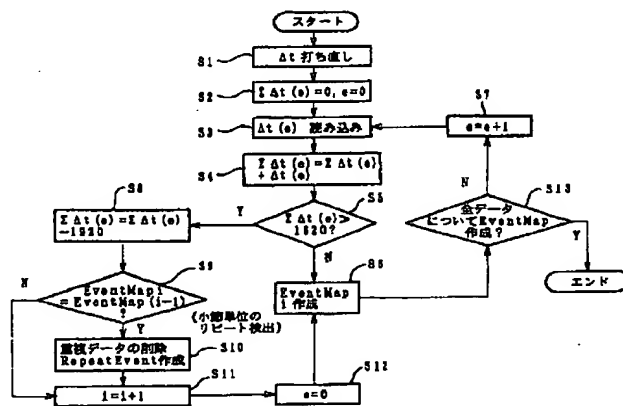
(74) 代理人 弁理士 二瓶 正敬

(54) 【発明の名称】 演奏情報圧縮方法及び演奏情報伸張方法

(57) 【要約】

【目的】 楽音を構成する音の音程と発音時間を指定する演奏情報のデータ量を大幅に削減すべく演奏情報を圧縮することのできる演奏情報圧縮方法と、かかる方法で圧縮された演奏情報を伸張して元の演奏情報を得ることのできる演奏情報伸張方法を提供する。

【構成】 演奏情報を所定時間単位のパート毎に分割し、パート毎の演奏情報を時系列順に比較し、時間軸方向の変化のパターンが2パート以上連続し、演奏情報に繰り返しがあるか否かを判断し、繰り返しがあると判断されたとき、繰り返しパート数を計算し、繰り返しのパートが重複しないよう削除し、削除される部分に、パターンの繰り返しが生じる旨を示す情報と、計数されたパターンの繰り返し回数を示す情報を付加する。再生時には、これらの情報を用いて圧縮された演奏情報を伸張する。他の実施例では複数パートにわたった演奏情報が繰り返される場合の圧縮と伸張が可能である。



【特許請求の範囲】

【請求項1】 楽音を構成する音の音程と発音時間を指定する演奏情報を受けて、前記演奏情報を所定時間単位のパート毎に分割するステップと、
前記分割するステップにて得られた前記パート毎の前記演奏情報を時系列順に比較し、前記演奏情報の時間軸方向の変化のパターンが2パート以上連続し、前記演奏情報に繰り返しがあるか否かを判断するステップと、
前記判断するステップにて繰り返しがあると判断されたとき、その繰り返しパート数を計数するステップと、
前記判断するステップにて繰り返しがあると判断されたとき、繰り返しのパートが重複しないよう削除するステップと、
前記判断するステップにて繰り返しがあると判断されたとき、前記削除する手段によって削除される部分に、パターンの繰り返しが生じる旨を示す情報と、前記計数するステップによって計数されたパターンの繰り返し回数を示す情報を付加するステップとを、
有する演奏情報圧縮方法。

【請求項2】 楽音を構成する音の音程と発音時間を指定する演奏情報を受けて、前記演奏情報を所定時間単位のパート毎に分割するステップと、
前記分割するステップにて得られた前記パート毎の前記演奏情報を時系列順に比較し、前記演奏情報の時間軸方向の変化のパターンが一部の演奏情報を除いて、2パート以上連続するとき、前記演奏情報に繰り返しがあると判断するステップと、
前記判断するステップにて繰り返しがあると判断されたとき、その繰り返しパート数を計数するステップと、
前記判断するステップにて繰り返しがあると判断されたとき、繰り返しのパート中、繰り返しではない異なる部分を残して、他の部分が重複しないよう削除するステップと、
前記判断するステップにて繰り返しがあると判断されたとき、前記削除するステップによって削除される部分に、パターンの繰り返しが生じる旨を示す情報と、前記計数するステップによって計数されたパターンの繰り返し回数を示す情報を付加するステップとを、
有する演奏情報圧縮方法。

【請求項3】 前記判断するステップが前記演奏情報の時間軸方向の変化のパターンを比較する際、音の音程と発音時間の照合において所定の誤差許容範囲を設け、比較精度を粗くするステップをさらに有する請求項1又は2記載の演奏情報圧縮方法。

【請求項4】 前記分割するステップにてパートに分割されて得られた前記各パート内において、所定の音程が連続し、繰り返されているか否かを判断する連続音程検出ステップと、
前記連続音程検出ステップにて繰り返しがあると判断されたとき、その繰り返し音程の情報出現回数を計数する

同一音程計数ステップと、

前記連続音程検出ステップにて繰り返しがあると判断されたとき、繰り返しの音程の情報が重複しないよう削除する音程情報削除ステップと、
前記連続音程検出ステップにて繰り返しがあると判断されたとき、前記音程情報削除ステップによって削除される部分に、音程の繰り返しが生じる旨を示す情報と、前記同一音程計数ステップによって計数された音程の繰り返し回数を示す情報を付加するステップとを、
さらに有する請求項1乃至3のいずれか1つに記載の演奏情報圧縮方法。

【請求項5】 楽音を構成する音の音程と発音時間を指定する演奏情報を受けて、前記演奏情報を所定時間単位のパート毎に分割するステップと、
前記分割するステップにて得られたパート列中の先頭のパートからパート単位で順次データ比較を行い、実質的に同一のデータのパートに同一パート番号を付与しファイルAを作成するステップと、
前記ファイルAの先頭からパート番号を調べ、重複する番号が出現したときは、それを削除した新たなファイルBを作成するステップと、
前記ファイルAとファイルBを先頭のパートからパート単位でパート番号の比較を行い、互いに相違するパート番号が出現したら、又は前記ファイルAのパートと比較する前記ファイルBのパートが存在しないときは、挿入すべきジャンプマークとタグの各々における識別のためのレベルをインクリメントし、前記ファイルB中の相違するパート番号の直前に前記レベルのインクリメントされたジャンプマークを挿入し、かつ前記ファイルA中の相違するパート番号と同一の前記ファイルB中のパート番号の直前に前記レベルのインクリメントされたタグを挿入するステップと、
前記ファイルA中の前記相違するパート番号以後と前記ファイルBの前記タグの挿入された直後のパート以後とをパート単位でパート番号の比較を行い、互いに相違するパート番号が出現したら、前記挿入するステップと同様の挿入を行うステップとを、
有する演奏情報圧縮方法。

【請求項6】 楽音を構成する音の音程と発音時間を指定する演奏情報を受けて、前記演奏情報を所定時間単位のパート毎に分割するステップと、
当初内容のないファイルCを作成するステップと、
前記ファイルC内の位置を示すファイルCのポインタを前記ファイルCの先頭に設定するステップと、
当初内容のないパートリストを作成するステップと、
変数iに前記ファイルAの最初のパート番号を設定するステップと、
前記変数iが前記パートリストにあるか否かを判断するパート番号確認ステップと、
前記パート番号確認ステップで前記変数iで示されるパ

ート番号がないときは、前記ポインタが前記ファイルCの末尾にあるか否かを判断するポインタ位置第1判断ステップと、
 前記ポインタ位置が前記ファイルCの末尾以外るときは、前記ポインタが現在ある位置にジャンプマークJUMP(n)を挿入し、次に前記ポインタを前記ファイルCの末尾へ移動させ、その後前記ポインタの位置にタグTAG(n)を挿入し、前記変数nを1つインクリメントするステップと、
 前記ポインタ位置が前記ファイルCの末尾であるときは、前記ファイルCに前記ファイルAの前記変数iで示されるパートの演奏情報を付加し、前記パートリストに前記変数iを付加し、前記ポインタを前記ファイルCの末尾に移動させるステップと、
 前記パート番号確認ステップで前記変数iで示されるパート番号があるときは、前記ポインタが前記ファイルCの変数iで示されるパートの先頭にあるか否かを判断するポインタ位置第2判断ステップと、
 前記ポインタ位置が前記ファイルCの変数iで示されるパートの先頭以外るときは、前記ポインタが現在ある位置にジャンプマークJUMP(n)を挿入し、次に前記ポインタを前記ファイルCの変数iで示されるパートの先頭位置へ移動させ、その後前記ポインタの位置にタグTAG(n)を挿入し、前記変数nを1つインクリメントするステップと、
 前記ポインタが前記ファイルCの変数iで示されるパートの先頭であるときは、前記ポインタを次のパートの先頭位置へ移動させるステップと、
 前記2つの移動させるステップ一方の後に、前記ファイルAの中の次のパートを検索し、存在する場合は前記変数iに前記ファイルAの次のパート番号を設定する次パート番号設定ステップとを、
 有し、以後前記ファイルAに次のパートがなくなるまで、前記パート番号確認ステップ以降、前記次パート番号設定ステップまでを繰り返し実行するようにした演奏情報圧縮方法。

【請求項7】 演奏情報の繰り返し部分を削除し、繰り返し部分を再生するために挿入されたジャンプマークとタグであってレベルの付されたものを含む圧縮演奏情報ファイルをメモリにロードするステップと、
 前記メモリにロードされた前記圧縮演奏情報ファイルについて前記タグを検出するステップと、
 前記メモリにロードされた前記圧縮演奏情報ファイルについて前記ジャンプマークをそのレベル順に検出するス*

* テップと、

前記ジャンプマークが検出されたとき、先に検出された前記タグのレベルと、その前記メモリのアドレスを指定して前記メモリからのデータの読み出し制御を行うステップとを有する演奏情報伸張方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は演奏情報の圧縮及び伸長に関し、特に楽音を構成する音の音程と発音時間を指定する演奏情報のデータ量を少なくするための圧縮と、一旦圧縮された演奏情報の伸長に関する。

【0002】

【従来の技術】楽音を構成する音の音程と発音時間を指定する演奏情報として例えばMIDI信号がある。MIDI信号はSMF(スタンダードMIDIファイル)という形式でメモリ等に蓄積されたり、伝送されたりする。かかるMIDI信号は従来の光ディスク等の記憶媒体に記録されたデータを読み出す方式のカラオケ装置に代って通信カラオケといわれているシステムに用いられている。

【0003】この通信カラオケはカラオケの音楽、歌詞、関連する画像(動画又は静止画)等を光ディスク等から出力された信号の情報の形態でそのまま送信すると、データ量が膨大であることから、楽音情報についてはMIDI信号としてステーションから各ユーザーの使用場所に設けた端末へ送信し、歌詞については文字コードとして送信するもので、必要に応じて所望の曲のデータを配信するものである。かかる例として株式会社エクスシング製の「JOYSOUND」システムが知られている。なお、音符、信号よりなるメロディの楽譜データについて繰り返し部分を効率よく処理する技術が特公昭61-91697号公報に示されているが、これはMIDI信号を前提とするものではなく、繰り返し等の記号を用いた楽譜が存在することを前提としたものであり、SMFで表現されるような既にある演奏情報を圧縮する技術ではない。さらに特開平4-147192号公報には楽譜の繰り返し部分をパターン化する技術が示されているが、十分なデータの圧縮は期待できない。

【0004】上記従来の通信カラオケシステムをはじめ、これまでに提案されたシステムは次の表1のように分類される。

【0005】

【表1】

ソース		データ量	音の品質
CD、LDを再生するもの (実際の演奏・コーラスが記録されている)		膨大	最高
MIDIに よるもの	演奏して入力したものにコーラスを付加	大	高
	演奏して入力(コーラス付加せず)	中	中
	音符から機械的に入力(コーラス付加せず)	小	低

【0006】

【発明が解決しようとする課題】表1からわかるように、MIDI信号を用いた場合、CD、LD等から再生された音の品質とは同一ではないが、品質を向上させようとするればデータ量が増大することがわかる。データ量が多いと、ISDNのような高速デジタル回線を用いなければリアルタイム通信が不可能となり、またデータの蓄積、保存の上でも必要メモリー容量が大きくなりコストの上昇につながってしまう。かかる問題は、通信カラオケシステムが一般家庭へ普及していくための障害となっている。また、通信上の問題とは別に、ハードディスク等の大容量メモリに多数の曲の演奏情報を蓄積しておく、所謂パッケージカラオケシステムにおいても、データ量の圧縮が求められている。

【0007】音楽は繰り返しの芸術であると言われるように、演奏が一定のパターンで繰り返されることが多い。ところがMIDI信号のような演奏情報では時間軸上に発生する次々のイベントの順列をそのままの形で扱っているため、一般的なデジタル信号の圧縮技術によるデータ量の圧縮を行うにとどまっていた。このため演奏情報を配信する通信カラオケやメモリに多数曲のデータを蓄積したパッケージ形式のカラオケシステム等では、データ量の圧縮・削減によるコスト低減、通信時間の短縮等が求められている。

【0008】したがって、本発明は楽音を構成する音の音程と発音時間を指定する演奏情報のデータ量を大幅に削減すべく演奏情報を圧縮することのできる演奏情報圧縮方法と、かかる方法で圧縮された演奏情報を伸張して基の演奏情報を得ることのできる演奏情報伸張方法を提供することを目的とする。

【0009】

【課題を解決するための手段】上記目的を達成するために本発明では、演奏情報が一定の時間単位で繰り返され、重複部分があることに着目し、かかる繰り返し部分を検出し、これを削除するとともに、再生時に圧縮情報を正しく再現できるよう、削除された繰り返し部分の位置等を示す信号を挿入するようにし、再生時には、この信号を用いて圧縮された演奏情報を伸張して、元の演奏情報を得るようにしたものである。

【0010】すなわち本発明によれば、楽音を構成する音の音程と発音時間を指定する演奏情報を受けて、前記演奏情報を所定時間単位のパート毎に分割するステップと、前記分割するステップにて得られた前記パート毎の前記演奏情報を時系列順に比較し、前記演奏情報の時間軸方向の変化のパターンが2パート以上連続し、前記演奏情報に繰り返しがあるか否かを判断するステップと、前記判断するステップにて繰り返しがあると判断されたとき、その繰り返しパート数を計数するステップと、前記判断するステップにて繰り返しがあると判断されたとき、繰り返しのパートが重複しないよう削除するステップと、前記判断するステップにて繰り返しがあると判断されたとき、前記削除する手段によって削除される部分に、パターンの繰り返しが生じる旨を示す情報と、前記計数するステップによって計数されたパターンの繰り返し回数を示す情報を付加するステップとを、有する演奏情報圧縮方法が提供される。

【0011】さらに本発明によれば、楽音を構成する音の音程と発音時間を指定する演奏情報を受けて、前記演奏情報を所定時間単位のパート毎に分割するステップと、前記分割するステップにて得られた前記パート毎の前記演奏情報を時系列順に比較し、前記演奏情報の時間軸方向の変化のパターンが一部の演奏情報を除いて、2パート以上連続するとき、前記演奏情報に繰り返しがあると判断するステップと、前記判断するステップにて繰り返しがあると判断されたとき、その繰り返しパート数を計数するステップと、前記判断するステップにて繰り返しがあると判断されたとき、繰り返しのパート中、繰り返しではない異なる部分を残して、他の部分が重複しないよう削除するステップと、前記判断するステップにて繰り返しがあると判断されたとき、前記削除するステップによって削除される部分に、パターンの繰り返しが生じる旨を示す情報と、前記計数するステップによって計数されたパターンの繰り返し回数を示す情報を付加するステップとを、有する演奏情報圧縮方法が提供される。

【0012】さらに本発明によれば、楽音を構成する音の音程と発音時間を指定する演奏情報を受けて、前記演奏情報を所定時間単位のパート毎に分割するステップ

20

30

40

50

と、前記分割するステップにて得られたパート列中の先頭のパートからパート単位で順次データ比較を行い、実質的に同一のデータのパートに同一パート番号を付与しファイルAを作成するステップと、前記ファイルAの先頭からパート番号を調べ、重複する番号が出現したときは、それを削除した新たなファイルBを作成するステップと、前記ファイルAとファイルBを先頭のパートからパート単位でパート番号の比較を行い、互いに相違するパート番号が出現したら、又は前記ファイルAのパートと比較する前記ファイルBのパートが存在しないときは、挿入すべきジャンプマークとタグの各々における識別のためのレベルをインクリメントし、前記ファイルB中の相違するパート番号の直前に前記レベルのインクリメントされたジャンプマークを挿入し、かつ前記ファイルA中の相違するパート番号と同一の前記ファイルB中のパート番号の直前に前記レベルのインクリメントされたタグを挿入するステップと、前記ファイルA中の前記相違するパート番号以後と前記ファイルBの前記タグの挿入された直後のパート以後とをパート単位でパート番号の比較を行い、互いに相違するパート番号が出現したら、前記挿入するステップと同様の挿入を行うステップとを、有する演奏情報圧縮方法が提供される。

【0013】さらに本発明によれば、楽音を構成する音の音程と発音時間を指定する演奏情報を受けて、前記演奏情報を所定時間単位のパート毎に分割するステップと、当初内容のないファイルCを作成するステップと、前記ファイルC内の位置を示すファイルCのポインタを前記ファイルCの先頭に設定するステップと、当初内容のないパートリストを作成するステップと、変数iに前記ファイルAの最初のパート番号を設定するステップと、前記変数iが前記パートリストにあるか否かを判断するパート番号確認ステップと、前記パート番号確認ステップで前記変数iで示されるパート番号がないときは、前記ポインタが前記ファイルCの末尾にあるか否かを判断するポインタ位置第1判断ステップと、前記ポインタ位置が前記ファイルCの末尾以外のときは、前記ポインタが現在ある位置にジャンプマークJUMP(n)を挿入し、次に前記ポインタを前記ファイルCの末尾へ移動させ、その後前記ポインタの位置にタグTAG

(n)を挿入し、前記変数nを1つインクリメントするステップと、前記ポインタ位置が前記ファイルCの末尾であるときは、前記ファイルCに前記ファイルAの前記変数iで示されるパートの演奏情報を付加し、前記パートリストに前記変数iを付加し、前記ポインタを前記ファイルCの末尾に移動させるステップと、前記パート番号確認ステップで前記変数iで示されるパート番号があるときは、前記ポインタが前記ファイルCの変数iで示されるパートの先頭にあるか否かを判断するポインタ位置第2判断ステップと、前記ポインタ位置が前記ファイルCの変数iで示されるパートの先頭以外のときは、前

記ポインタが現在ある位置にジャンプマークJUMP

(n)を挿入し、次に前記ポインタを前記ファイルCの変数iで示されるパートの先頭位置へ移動させ、その後前記ポインタの位置にタグTAG(n)を挿入し、前記変数nを1つインクリメントするステップと、前記ポインタが前記ファイルCの変数iで示されるパートの先頭であるときは、前記ポインタを次のパートの先頭位置へ移動させるステップと、前記2つの移動させるステップ一方の後に、前記ファイルAの中の次のパートを検索し、存在する場合は前記変数iに前記ファイルAの次のパート番号を設定する次パート番号設定ステップとを、有し、以後前記ファイルAに次のパートがなくなるまで、前記パート番号確認ステップ以降、前記次パート番号設定ステップまでを繰り返し実行するようにした演奏情報圧縮方法が提供される。

【0014】さらに本発明によれば、演奏情報の繰り返し部分を削除し、繰り返し部分を再生するために挿入されたジャンプマークとタグであってレベルの付されたものを含む圧縮演奏情報ファイルをメモリにロードするステップと、前記メモリにロードされた前記圧縮演奏情報ファイルについて前記タグを検出するステップと、前記メモリにロードされた前記圧縮演奏情報ファイルについて前記ジャンプマークをそのレベル順に検出するステップと、前記ジャンプマークが検出されたとき、先に検出された前記タグのレベルと、その前記メモリのアドレスを指定して前記メモリからのデータの読み出し制御を行うステップとを有する演奏情報伸張方法が提供される。

【0015】

【実施例】以下図面と共に本発明の好ましい実施例について説明する。図1は本発明の演奏情報圧縮方法の第1実施例を示すフローチャートである。図1のフローチャートについて説明する前に、本発明の適用される例として通信カラオケシステムについて図2、図3を用いて説明する。図2は本発明の演奏情報圧縮方法により、カラオケ情報としてのMIDI情報を圧縮してステーション(ホスト)から各端末へ送信する通信カラオケシステムの一例を示す模式ブロック図である。図3はステーションが圧縮しないデータと、本発明の演奏情報圧縮方法により事前に圧縮してあるデータの双方を送信/蓄積ファイルA、Bの2種類用意しておき、端末や回線の能力に応じて2つのファイルを使い分ける構成例を示す模式ブロック図である。どちらのファイルから読み出して送信するかは判断は、通信回線(ネットワーク)がISDN等の高速回線か一般公衆回線かを判断したり、端末側の伸張能力を検出することにより行う。なお、図2、図3いずれのシステムにおいても、端末側のハードディスク等のメモリにパッケージとして予め数千曲分のカラオケ情報を本発明の演奏情報圧縮方法により圧縮したMIDI情報として蓄積しておくことができる。この場合、新譜やリクエスト頻度の低い曲で端末のメモリに蓄積され

ていない曲の情報を通信回線を用いてステーションから端末へ送信することができる。

【0016】図4は1つの曲の演奏情報を記述したSMFを圧縮して圧縮ファイルを作成する手順の全体概念を示すブロック図である。多数曲分のSMFは例えばハードディスク等のメモリに記憶されており、この内、図示しない制御系により指定された1曲分のSMF1がMIDI CH（チャンネル）分離ブロック2にてチャンネル毎に分離される。すなわち演奏すべき楽器（音源）に対応するチャンネル毎に以下の処理は行われる。ブロック3は $(\Delta t + event)$ ブロックリピートを検出するものである。すなわち、後述する演奏情報のパターンに繰り返し（リピート）があるとき、これを検出するものである。次のブロック4はR（リピート）符号化処理を行うもので、ブロック3で検出された重複部分について重複しないよう削除して、かつ必要な信号を付加する処理が行われる。次のマージ処理ブロック5は多チャンネル分のR符号化処理の行われた信号を元のSMF形式のように合体するものであり、その結果 Δt （デルタタイム）が一歩化されて、圧縮ファイル6が作られる。

【0017】図5はSMFによる記述から繰り返し部分を検出し、重複部分を削除して、圧縮した結果得られる圧縮ファイルを作成する場合のデータ列の変化態様を示す図である。図5中“EVENT”はMIDI信号の“イベント”を示し、“T”は“ Δt ”と同様の時間情報を、“R”は“リピート（繰り返し）”を示している。図6は図5に示した情報“T”又は“R”のフォーマットを示している。このデータは図示のように8ビット構成であり、各ビットは図示の内容を示す。図7は1つの例として、繰り返し部分のある楽譜を示している。この例では第2小節と第3小節が同一のパターンとなっている。

【0018】ここで図1のフローチャートに戻って本発明の演奏情報圧縮方法の第1実施例について説明する。このフローは図4で示したブロック3とブロック4に相当する。ステップS1にてチャンネル単位で先頭から Δt で表記し、ステップS2で $\Delta t(e)$ の累計である $\Sigma \Delta t(e)$ と個々のイベントを示す変数 e をそれぞれ0にイニシャライズする。ステップS3では $\Delta t(e)$ を読み込み、ステップS4では $\Delta t(e)$ だけインクリメントし、ステップS5では $\Sigma \Delta t(e)$ が1920を超えたか否かを判断する。この数値1920は4分音符1個あたりのティクス（Ticks）数を480とし、1小節が4分音符4個分の長さ（すなわち拍子記号が4/4）とした場合の1小節の長さに相当する。なおTickはSMFにおける最小時間単位である。このステップS5は演奏情報であるSMFを所定時間単位のパート（この例では1小節）毎に分割するものである。ステップS5で1小節が経過するまでの間にステップS6を繰り返し実行し、イベントマップ（EVENT MAP）

iを作成する。イベントマップiはパート毎のイベントの集合である。ステップS7では e を1つインクリメントする。1小節が終了するとステップS8で Δt の累計から1920を減じ、次のステップS9で前の小節（i-1）のイベントマップ（i-1）と、現在の小節iのイベントマップiの内容が一致するか否かを判断する。一致しないときはステップS11でiを1つインクリメントする、一致するときはステップS10で重複データの後ろの方を削除し、リピートイベント（Repeat Event）を作成する。リピートイベントは図5に示す“R”に相当し、図6のデータフォーマットを有する。図6で $(ID1, ID0) = (0, 0)$ 及び $(ID1, ID0) = (0, 1)$ の場合がT、 $(ID1, ID0) = (1, 0)$ 及び $(ID1, ID0) = (1, 1)$ の場合がRである。単純リピート及び区間指定リピートはチャンネル単位で行い、チャンネルはリピートバイトの直前のイベントに基づいて決定される。図6の $-\Delta ts$ はマイナス $\Delta t - start$ 、 $-\Delta te$ はマイナス $\Delta t - end$ の意味であり、 $-\Delta ts$ 及び $-\Delta te$ はマイナスを取って絶対値で表現している。なお、ステップS10では、リピート検出回数を計数し、図6中のリピートイベント中のリピート回数として記述する。ステップS10の後にはステップS11が実行される。ステップS11の後にはステップS12にて e を0としてステップS6へ行く。なお、ステップS13では全データについてイベントマップが作成されたか否かを判断し、YESならこのフローを終了する。

【0019】図1のフローによって図7の譜例に示される曲の演奏情報を処理すると、イベントマップ同志の比較から、同一の音程、同一の発音時間を示すパートの繰り返しとして、第2小節と第3小節が検出される。図1の実施例では演奏情報の時間軸方向の変化のパターンが2パート以上連続したとき、すなわちイベントマップの内容が完全に一致したとき、繰り返しがあると判断しているが、他の例として、部分的な一致を検出したときも、繰り返しとみなし、異なる部分はそのまま記述するようにすることもできる。

【0020】さらに、パート同志の一致の有無の検出精度を粗くし、すなわち音程と発音時間の照合において所定の誤差許容範囲を設けるようにしてもよい。このようにすることにより、わずかなノイズや演奏ミスにより、本来繰り返し部分であるのに、非同一パターンと認識されてしまうことを防止でき、結果として繰り返しと認識されるパート数を増加させることができ、圧縮効率を高めることができる。さらに、演奏上の時間軸のゆらぎ、あるいは演奏の強さのゆらぎなどの許容範囲を演奏再現時（再生時）にあまり影響を与えない程度に広げることにより、圧縮効率を高めることもできる。また、パート同志の比較の他に、同一パート内で所定の音程が繰り返し出現するような場合、例えばドラムなどの打楽器の演

奏情報の場合、これを検出して重複しないよう削除し、図1で示した処理と同様の処理にて音程単位で記述することもできる。本発明では所定時間単位のパート毎の検出が行われるが、この1パートは必ずしも1小節と同一である必要はなく、その数分の1としたり、数倍とすることもできる。さらに、パートの時間を2種類以上設け、例えば1小節単位の検出を行いつつ、その2分の1の時間単位でも繰り返しの検出を行うようにすることもできる。

【0021】図1に示した第1実施例は、連続するパートの内容が一致している場合に繰り返しであると判断しているが、いくつかの小節にわたったメロディーが繰り返される場合も多い。このような場合にも圧縮可能な手法について第2実施例と第3実施例で説明する。図23は圧縮前のSMFで8つのパートからなるものを模式的に示した図である。図23において、第2番目乃至第4番目のパートと第5番目乃至第7番目のパートが重複している。したがって冗長性を除くために、考えられる1つの手法として、重複する3つのパートを削除すると図24に示すように5つのパートのみとなる。この場合、「先頭から(t1+t2+t3+t4)の時間だけ再生したら、先頭からt1経過時点までジャンプする」という内容を記述しておくことになる。すなわち各パートの Δt の総和をそれぞれ計算しておき、再生開始時点からの経過時間によって再生位置を再設定するような記述を設けている。しかし、この方法によると、パートの再編集の必要性が生じた場合、記述部を同様に編集しなくてはならない。また記述が複雑であるほど時間の再計算も複雑化してしまう。

【0022】前述の特開昭61-91697号公報では、楽譜の省略記号をコード化し、音符情報と共に記憶し、再生時に省略記号に従って制御を行うようにしている。これによれば冗長性の問題と再編集の問題は解決するものの、現代の楽曲は構成が複雑化しており、省略記号を単にコード化しただけでは楽曲構成を正しく表記できない場合が生じてくる。図25に示した例のように、D. S. (ダルセーニョ)によって戻って再度演奏する場合に繰り返しを省くことがよく行われるが、このように注釈によって指示される選択的な演奏情報を表記することは上記公報の技術では不可能である。また、上記公報の技術は前述のように繰り返し等の記号を用いた楽譜を前提としている。

【0023】したがって本発明の演奏情報圧縮方法の第2実施例及び第3実施例では、複数回重複して記述される同一パートを含み直列状に記述されている演奏情報に対して、演奏情報と識別可能なジャンプマークとタグを設け、楽曲の繰り返しを示すことによって、少ない情報量で記述するようにしている。また、ジャンプマークに条件値(レベル)を含めることによって複雑な演奏情報を記述することができる。

【0024】図13はジャンプマークとタグの構成を示す図である。いずれも、非演奏情報ID、チャンネルID、レベルIDを有している。レベルIDは同一マークの相互間の識別を行うか、又は順序を示すものである。なお第2実施例と第3実施例は、共に複数のパート(各実施例では小節)にわたる演奏情報が繰り返されるときに、繰り返しを削除してジャンプマークとタグを挿入し、結果として同じ圧縮ファイルを作るものである。第2実施例は元のファイルAの他に重複のないファイルBを一旦作成する必要がある点で後述する第3実施例より手順が多い。しかし、圧縮の手順を理解するためには適当であるので、まず第2実施例について説明し、その後、より利用価値の高い第3実施例について説明する。図14と図15は同一パートの検出と削除の概念を示す図である。図14の1部の楽譜に示すように7小節と省略記号で記述された曲を展開すると図14の下部に示すようなものとなる。なお丸印の中の数字は便宜的に記入したものであって、どの小節とどの小節の演奏情報が同一であるのかは、この時点ではわかっていない。

【0025】図15は図14の下部に示されたような演奏情報があるとき、このデータ列から重複部分を検出する手法を示す図である。演奏情報は小節単位で時間軸上に図15の上段に示すように並んでいる。ここで先頭の小節xから以降の小節に対してデータの比較を行う。一致したなら若い順にパート番号(この場合1)をふる。演奏情報の作成がパートのコピーによって行われない場合は時間軸方向の揺らぎ等が含まれるため、データのファイリングやクォンタイズ、認識的手法が用いられる。このようにしてパート番号のふられた情報を納めたファイルをA、重複したパートを削除した情報を納めたファイルをBとする。図16はこの2つのファイルA、Bと本発明の演奏情報圧縮方法の第2実施例により最終的に作成される圧縮ファイルを示している。両方のファイルA、Bを読み進め、異なるパートが出現したら、レベルnをインクリメントしファイルBにおいて、ジャンプマークJ(n)を挿入しファイルAの次に出現するパートに戻ってタグT(n)を挿入する。

【0026】レベルnはジャンプマークJ及びタグTのそれぞれにおいて相互識別をするためのものであり、自然数で1、2、3…とインクリメントする。図16の下方には、ジャンプマークJとタグTを挿入する手法が示されている。すなわち、ファイルAとファイルBを先頭のパートからパート単位でパート番号の比較を行い、互いに相違するパート番号が出現したら、又はファイルAのパートと比較するファイルBのパートが存在しないときは、挿入すべきジャンプマークとタグの各々における識別のためのレベルnをインクリメントし、ファイルB中の相違するパート番号の直前にレベルのインクリメントされたジャンプマークを挿入し、かつ前記ファイルA中の相違するパート番号と同一のファイルB中のパート

番号の直前にレベルのインクリメントされたタグを挿入する。ファイルA中の相違するパート番号以後とファイルBのタグの挿入された直後のパート番号とをパート単位でパート番号の比較を行い、互いに相違するパート番号が出現したら、上記の挿入動作と同様の挿入が行われる。

【0027】図17及び図18は上記第2実施例を用いて複雑な繰り返しのある曲の演奏情報を圧縮することができる例を示す図であり、図17はかかる複雑構成の曲の例を示し、図18は圧縮前のファイルA、Bと圧縮ファイルを示す図である。

【0028】ここでMIDI信号情報ファイルとしてのSMFとSMFを変形して上記ジャンプマーク、タグを挿入してゆく手法について説明する。SMFは以下のものを含む。

- ・MIDIイベント : 演奏情報
- ・METAイベント : 非演奏情報

どちらのイベントも時間情報 Δt を持っているので、META(メタ)イベントは、MIDIイベントとの相対的な時間的位置を指定して、挿入することができる。また、METAイベントは、これが持つデータフォーマットを規定することができる。通常のSMFでは、パートの概念はないが、パートの先頭であるという識別子(PART ID)を持ったメタイベントを定義することによって、SMF中にパート番号を記述することができる。

【0029】次にパートの定義について説明する。例えば1小節に相当する各々のパートは図15に示されるようにパート番号により指定される。したがって各パートはMETAイベントから始る、SMF中の一連のMIDIイベント群である。このMETAイベントを、META(PART, x)と略記する。

META(PART, x) $\rightarrow \Delta t$: META ID: PART ID: PARTNO x とする。比較のために、MIDIイベントのNote on(打鍵)は、Note on $\rightarrow \Delta t$: MIDI Note on ID: MIDIチャンネル: 音程: 音量
SMFの構成は図19に示すようになっている。

【0030】META(PART, x)の挿入位置は、自由であるが、繰り返しの検出に用いるので、繰り返しが予想される単位以下で挿入するのが良い。例えば、図20に示すような楽譜をもとにSMFを作成する場合、作成者は、“A, B, C, D, E”それぞれのMIDI演奏情報を作成し、実際の演奏手順に従って、データのコピーペースト等の手法で“A B C D A B C E”となったMIDI演奏情報列をSMFに収めるが、楽譜上の繰り返し記号と、META(PART, x)の入れ方の規約を作ることによって、図21に示すようなSMFを作成することが可能である。

【0031】あるいは作成者が以上のような、作成手順

に全く関知しない通常のSMFを作成する場合、繰り返し部分を自動的に検出するためには、SMFを受け取った側で1小節=1パートとなるように、META(PART, x)を挿入するのが良い。SMF中の小節の切れ目は既にSMFフォーマット中に定められている。META(time signature)等の情報を、SMFから読み出すことによって、簡単に知ることができる。前出の例によれば、META(PART, x)が挿入されたらSMFは図22に示すようになる。どちらの場合でも、繰り返し部分の検出は問題なく行われる。また、META(PART, x)は、検出のために挿入されたものであるから、最終的に圧縮記述されたファイル中に存在していなくても演奏の再生に対して、全く問題がない。

【0032】次に本発明の演奏情報圧縮方法の第3実施例について説明する。前述の手順によりMETA(PART, x)が挿入されたSMFをSMF Aとし、SMF Aを圧縮した圧縮ファイルをSMF Cとする。なお、この圧縮ファイルSMF Cは、スタンダードMIDIファイルの形式とは異なるので、SMFと呼ぶことは必ずしも適当ではないが、ここでは便宜的にSMF Cとしておく。

【0033】次にいくつかの定義を行っておく。(a) PART LIST(パートリスト): パート番号の列である。

(b) f, C(ファイルポインタ): 圧縮ファイルSMF Cの内容の任意の位置を差し示すもの。

【0034】ジャンプマークJとタグTはMETAイベントを用いて次のように記述する。

META(JUMP, Ch, y)

META(TAG, Ch, z)

【0035】これらを一組として圧縮ファイルSMF Cに次のように記述する。

META(JUMP, Ch, y) $\rightarrow \Delta t$: META ID: JUMP ID: MIDIチャンネル Ch: レベルy

META(TAG, Ch, z) $\rightarrow \Delta t$: META ID: TAG ID: MIDIチャンネル Ch: レベルz

【0036】なお、ジャンプマークとタグのそれぞれにレベルを付加することによって、複雑な演奏手順に対しても、統一的な書式で記述し、正しく再生することができる。

【0037】ここで、MIDIチャンネルについて説明する。MIDIでは、複数の楽器が同時に演奏されている情報を伝送できるように、MIDIのイベントに対して、MIDIチャンネル(Ch)が付加されている。例えば、ピアノに関するイベントはCh=1、ベースはCh=2とし、受信側ではチャンネルによってイベントを分離し、それぞれを適切な音源へ転送することがなされ

ている。この概念は、SMFにおいても全く同様である。楽器毎には、別の演奏を行うも、SMF上では、混在しているため繰り返しの検出においてはSMF A中のイベントをチャンネル毎に分離し、各チャンネル独立に圧縮記述を行い、この後に、最終的なSMF Cへ、合成しながら、記録することが望ましい。合成後にも、各チャンネルのジャンプマークとタグが識別できるように、各イベントにMIDIチャンネルを持つのである。本明細書では説明の簡略のために、SMF Aは単一のチャンネルのイベントのみを含むものとして説明している。

【0038】次に第3実施例の圧縮手順についてフローチャートを示す図8及び図9を用いて説明する。圧縮手順のフローには2つのパスがある。1つめのパスは図8に示すように、SMF A中の同一パートを検出し、META (PART, x) の書換を行うものである。2つのパートを同じものと見なすために、比較を行う段階でMIDIフィルタリング、MIDIクオンタイズを用いてもよい。これは、SMFの内容が演奏者の実演奏等を記録したものであるとき、特に効果がある。すなわち、同一パートを演奏しようとした場合でも、実演奏では全く同じMIDIイベント群にはなり得ないため、MIDIフィルタリング、MIDIクオンタイズを行い、同一パートの検出率を向上させ、ひいては圧縮率を向上させるものである。MIDIフィルタリングは、演奏者が意図しない、極めて短い音に対応するMIDIイベントを削除すること等である。MIDIクオンタイズは発音の時間的な揺れを、大きなタイムスロットに量子化し直すことによって、検出率を向上させるものである。また、音量に対しても同様のことが適用できる。上記は、比較の際に用いる手法であり、最終的に生成されるファイルの内容に対しては用いられないのは当然である。2つ目のパスは図9に示すように、SMF Aから、SMF Cを生成するものである。

【0039】図8中“j++” “i++”は変数“j” “i”をインクリメントするステップである。変数iは比較する元のパートを示し、変数jは比較する対象のパートを示している。このフローではパートiとパートjの内容が一致したとき、繰り返しがあると判断しMETA (PART, j) をMETA (PART, i) に変更している。

【0040】次に図9に従って具体的圧縮手法について説明する。この第3実施例では先の第2実施例で説明した図16に示すファイルBに相当するものは用いる必要がない。すなわち、図16のファイルAに相当するSMF AからファイルBを用いることなく最終的に圧縮された圧縮ファイルSMF Cを得るものである。

【0041】ステップS21で当初内容のないファイルC (SMF C) を作成し、ファイルC内の位置を示すファイルCのポインタをファイルCの先頭に設定し、当

初内容のないパートリストを作成する。次のステップS22で変数iにファイルAの最初のパート番号を設定し、次のステップS23で変数iが前記パートリストにあるか否かを判断する。変数iで示されるパート番号がないときは、ポインタf_pCがファイルCの末尾にあるか否かをステップS24で判断し、ポインタf_pC位置が前記ファイルCの末尾以外のときは、ステップS25でポインタf_pCが現在ある位置にジャンプマーク

(n) を挿入し、次にポインタf_pCをファイルCの末尾へ移動させ、その後ポインタf_pCの位置にタグTAG (n) を挿入し、変数nを1つインクリメントする。ポインタf_pC位置がファイルCの末尾であるときは、ステップS26でファイルCに前記ファイルAの変数iで示されるパートの情報を付加し、パートリストに変数iを付加し、ポインタを前記ファイルCの末尾に移動させる。

【0042】パート番号の確認ステップで変数iで示されるパート番号があるときは、ポインタf_pCがファイルCの変数iで示されるパートの先頭にあるか否かをステップS29で判断する。次のステップS30でポインタf_pCの位置が前記ファイルCの変数iで示されるパートの先頭以外のときは、ポインタf_pCが現在ある位置にジャンプマークJUMP (n) を挿入し、次にポインタf_pCをファイルCの変数iで示されるパートの先頭位置へ移動させ、その後ポインタf_pCの位置にタグTAG (n) を挿入し、変数nを1つインクリメントする。ポインタf_pCが前記ファイルCの変数iで示されるパートの先頭であるときは、ステップS31でポインタf_pCを次のパートの先頭位置へ移動させる。この2つの移動させるステップS28又はS31の一方の後、ステップS32でファイルAの中の次のパートを検索し、存在する場合は、変数iにファイルAの次のパート番号を設定する。以後、ファイルA中に次のパートがなくなるまで、パート番号確認ステップS23以降、次パート番号設定ステップS32までを繰り返し実行する。

【0043】図10は図8に示すパス1の開始前のSMF Aと、その終了後に得られるSMF Aを、さらに図9に示すパス2の終了後に得られるSMF Cを示す図である。図中、A、B、C…は各パートの内容を示し、数字1、2、3…はパート番号を示している。またSMF Cはジャンプマークとタグの付加された様子を示している。

【0044】次に上記各実施例により圧縮された演奏情報を、例えば通信カラオケの端末側等で再生するために伸張する方法について説明する。図11は本発明の演奏情報伸張方法を実現するために用いられる演奏情報再生装置 (伸張装置) の実施例を示すブロック図である。記憶装置10は例えば大容量ハードディスクで、この中に図10に示す圧縮ファイルSMF Cが多数格納されて

いるものとする。このうち所望の曲の SMF C が図示しない制御装置のアドレス指示に従って読み出され、メモリ 12 にロードされる。この後、再生に先立ち、SMF C 中のタグのサーチが行われる。すなわち、リーダー 14 は、アドレスカウンタ 16 の指示に従ってメモリ 12 の内容を読んで行き、タグのサーチを行う。分離器 18 は、リーダー 14 の出力信号、すなわち SMF C から MIDI イベント、ジャンプレベル値、タグレベル値を分離抽出するものである。サーチの結果、タグを検出する毎に、タグアドレスメモリ 26 は分離器 18 から与えられるタグレベル値と、アドレスカウンタ 16 から与えられるアドレスを関係付けて図 12 に示すテーブルのような形で格納して行く。こうして、1 つの SMF C を全部サーチして図 12 に示すテーブルを完成させる。

【0045】その後、再生を開始するために、まずアドレスカウンタ 16 とレベルカウンタ 22 を初期化する。再生時には、分離器 18 からのタグレベル値はタグアドレスメモリ 26 に書込まない（又は分離器 18 からタグレベル値を出力しない）。分離器 18 から出力される MIDI イベントは各対応する（チャンネル毎の）図示しない音源に与えられる。分離器 18 の出力信号の 1 つとしてレベル比較器 20 にはジャンプレベル値が与えられる。レベルカウンタ 22 はそのカウント値をレベル比較器 20 へ出力し、レベル比較器 20 は 2 つの入力レベルを比較し、一致するとレベルカウンタ 22 を 1 つインクリメントさせる信号を出力するとともにゲート 24 へゲート信号を与える。ゲート 24 はレベルカウンタ 22 の出力カウント値、すなわち、検出されたジャンプマークのレベル値（一致レベル値）をゲート信号を受けたとき、選択器 28 へ与える。選択器 28 はタグアドレスメモリ 26 からジャンプレベルに対応したタグのアドレスを読み出して、アドレスカウンタ 16 にロードする。アドレスカウンタ 16 は、ロードされたアドレスに従って順次メモリ 12 内のデータを読み出して行く。

【0046】このようにして、メモリ 12 内の 1 つの SMF C の内容を読みながら MIDI イベントを順次出力することにより伸張動作が行われ、SMF C の全てのデータを読み出すことにより一曲分の再生を終了する。

【0047】図 11 に示した構成の上記説明では、再生に先立ってタグの位置をサーチして、そのアドレスを予めロードする方法を用いているが、伸張装置の処理能力が十分なものであれば、予めタグを読み込まず、タグをサーチしながら再生を同時に行うようにすることもできる。かかる構成の場合は、図 11 に示したタグアドレスメモリ 26 は不要となる。

【0048】

【発明の効果】以上詳細に説明したように、本発明の演奏情報圧縮方法によれば、MIDI 信号のように楽音を

構成する音の音程と発音時間を指定する演奏情報のデータ量を、メロディーの繰り返しを削除することにより、大幅に削減することができ、また本発明の演奏情報伸張方法によれば、かかる削減により圧縮された演奏情報を伸張して元の繰り返しのある演奏情報を再生することができ、よって通信コストの削減、メモリ容量の問題の解消がなされる。また、通信カラオケシステム等において、画像情報や文字情報を本来の演奏情報信号とともに送信したり、メモリに蓄積する場合に、演奏情報のデータ量が削減できるので、これらの付加的情報の取り扱いが容易となる。

【図面の簡単な説明】

【図 1】本発明の演奏情報圧縮方法の第 1 実施例を示すフローチャートである。

【図 2】本発明の演奏情報圧縮方法及び演奏情報伸張方法の適用される一例としての通信カラオケシステムのブロック図である。

【図 3】本発明の演奏情報圧縮方法及び演奏情報伸張方法の適用される一例としての通信カラオケシステムの他の構成例を示すブロック図である。

【図 4】本発明の演奏情報圧縮方法の概念を示すブロック図である。

【図 5】本発明の演奏情報圧縮方法の手法を説明する図である。

【図 6】本発明の演奏情報圧縮方法の第 1 実施例で用いられる信号のフォーマット図である。

【図 7】本発明の演奏情報圧縮方法の第 1 実施例で圧縮しようとする曲の一部の楽譜である。

【図 8】本発明の演奏情報圧縮方法の第 3 実施例の一部を示すフローチャートである。

【図 9】本発明の演奏情報圧縮方法の第 3 実施例の他の一部を示すフローチャートである。

【図 10】本発明の演奏情報圧縮方法の第 3 実施例における圧縮ファイル作成過程を説明する図である。

【図 11】本発明の演奏情報伸張方法を実現する再生器の一例を示すブロック図である。

【図 12】図 11 におけるタグアドレスメモリの内容を示す図である。

【図 13】本発明の演奏情報圧縮方法の第 2 実施例におけるジャンプマークとタグのデータ構成を示す図である。

【図 14】本発明の演奏情報圧縮方法の第 2 実施例の圧縮手順を説明する図である。

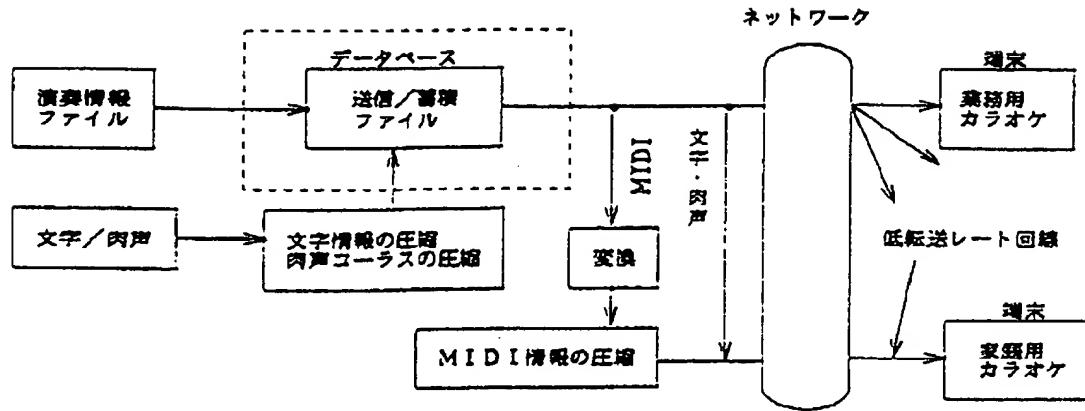
【図 15】本発明の演奏情報圧縮方法の第 2 実施例の圧縮手順を説明する図である。

【図 16】本発明の演奏情報圧縮方法の第 2 実施例の圧縮手順を説明する図である。

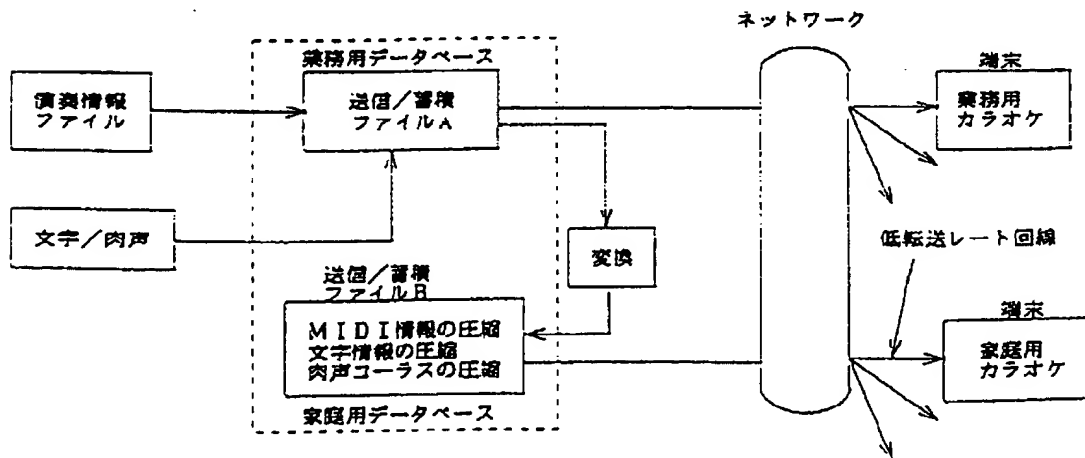
【図 17】本発明の演奏情報圧縮方法の第 2 実施例の圧縮手順を説明する図である。

【図 18】本発明の演奏情報圧縮方法の第 2 実施例の圧

【図2】

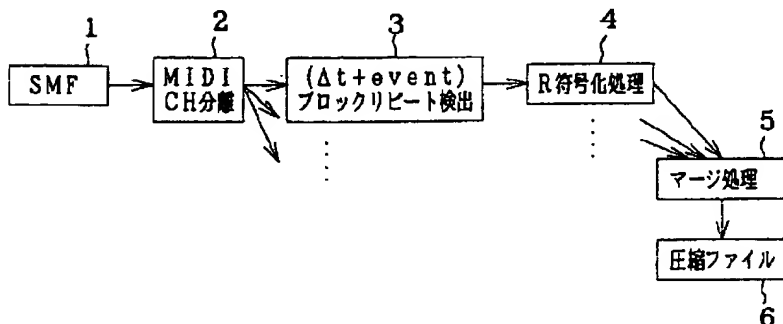


【図3】

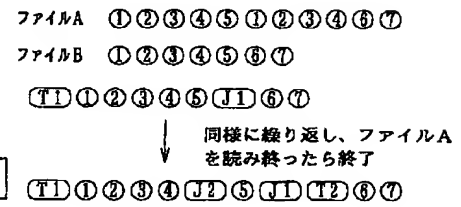
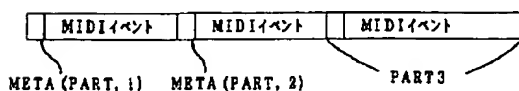


【図4】

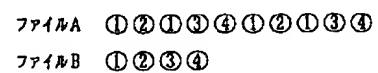
【図16】



【図19】



【図18】



【図5】

SMFによる記述

Δt_1 EVENT1 Δt_2 EVENT2 Δt_1 EVENT1 Δt_2 EVENT2 Δt_3 EVENT

(前の2バイトと同一演奏の繰返し)



本圧縮法方式による記述

T (Δt_1 と同定義) EVENT1 T (Δt_2 と同定義) EVENT2 R T (Δt_3 と同定義) EVENT3

(repeat)

【図6】

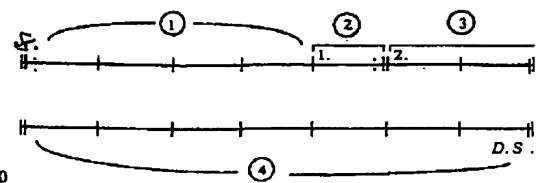
【図17】

T又はRのフォーマットデータ構造(SMFの Δt 形式を変更する)

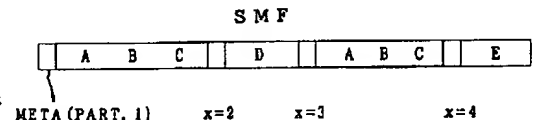
T又はR

ID1	ID0	D5	D4	D3	D2	D1	D0
-----	-----	----	----	----	----	----	----

- (ID1, ID0) = (0, 0) : Δt (従来と同じ)
 D5 : 後続バイトフラグ
 D4~D0 : Ticks 値(デルタタイムの最上位バイト(00-3F。SMFの場合は00-7F)を記述。後続バイトがある場合(D5=1の場合)には、2バイト目は40~1FFF(SMFの場合は80~3FFF)となる)
- (ID1, ID0) = (0, 1) : 拡張 Δt (指定された繰返し回のみ演奏を行う Δt)
 D5~D1 : 演奏回指定(その回のみ演奏)
 D0 : 後続バイトフラグ(0ならば $\Delta t=0$ とする)
- (ID1, ID0) = (1, 0) : 単純リピート(直前の指定区間を繰返し)
 D5~D0 : リピート回数(何回繰返すかを指定)
- (ID1, ID0) = (1, 1) : 区間指定リピート(指定された区間を繰返し)
 第2バイト : Δt の値(現在位置から何Ticks前までを繰返すかを指定)
 D5~D0 : 区間指定リピート(指定された区間を繰返し)
 第2バイト : リピート回数(何回繰返すかを指定(1回~64回))
 第3バイト : Δts の値(現在位置から何Ticks前から繰返すかを指定。
 第2バイトは可変長のため、 Δts は3バイト目に記述されるとは限らない)

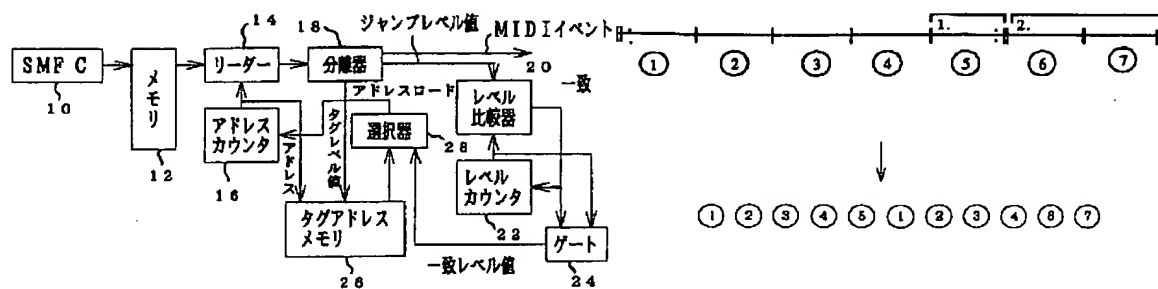


【図21】



【図11】

【図14】



【図13】

ジャンプマーク

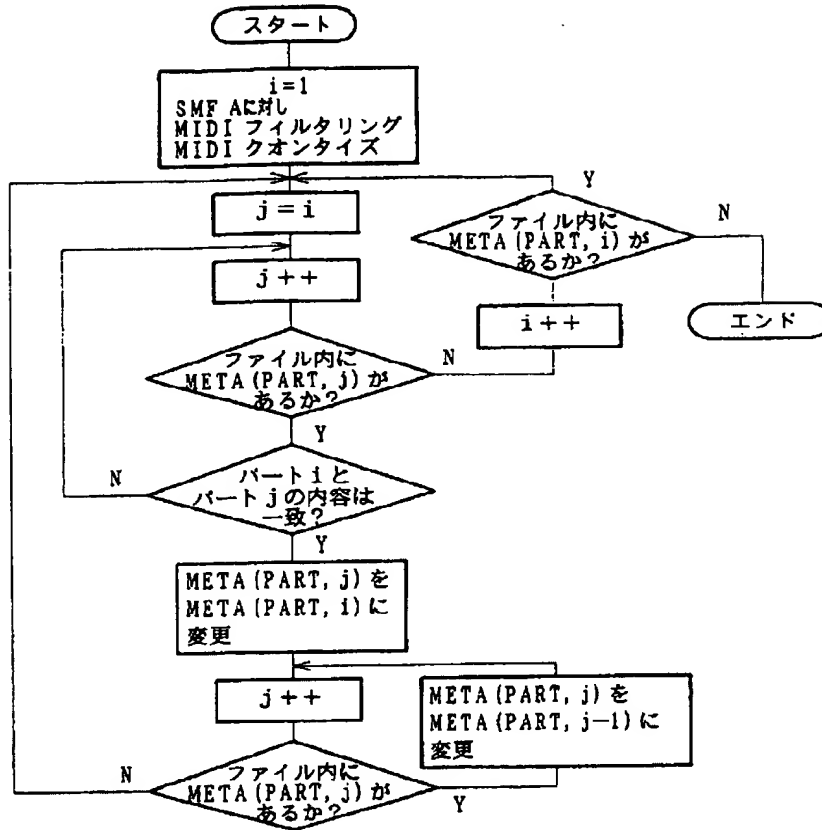
非演奏情報ID	マークID	チャンネルID	レベルID
---------	-------	---------	-------

タグ

非演奏情報ID	タグID	チャンネルID	レベルID
---------	------	---------	-------

【図8】

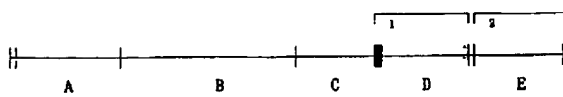
パス1
〈SMF Aの同一パートの検出〉



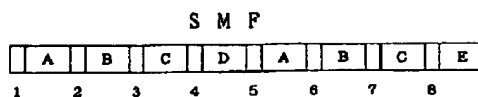
【図24】

各パートの時間	
パート1(イントロ)	t1
パート2(A)	t2
パート3(B)	t3
パート4(C)	t4
パート5(エンディング)	t5

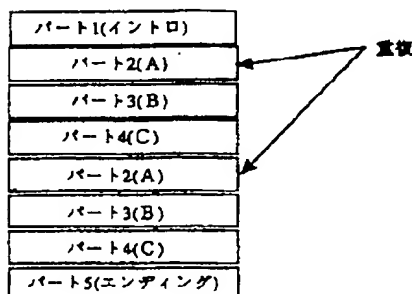
【図20】



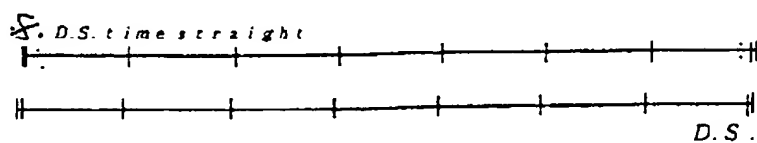
【図22】



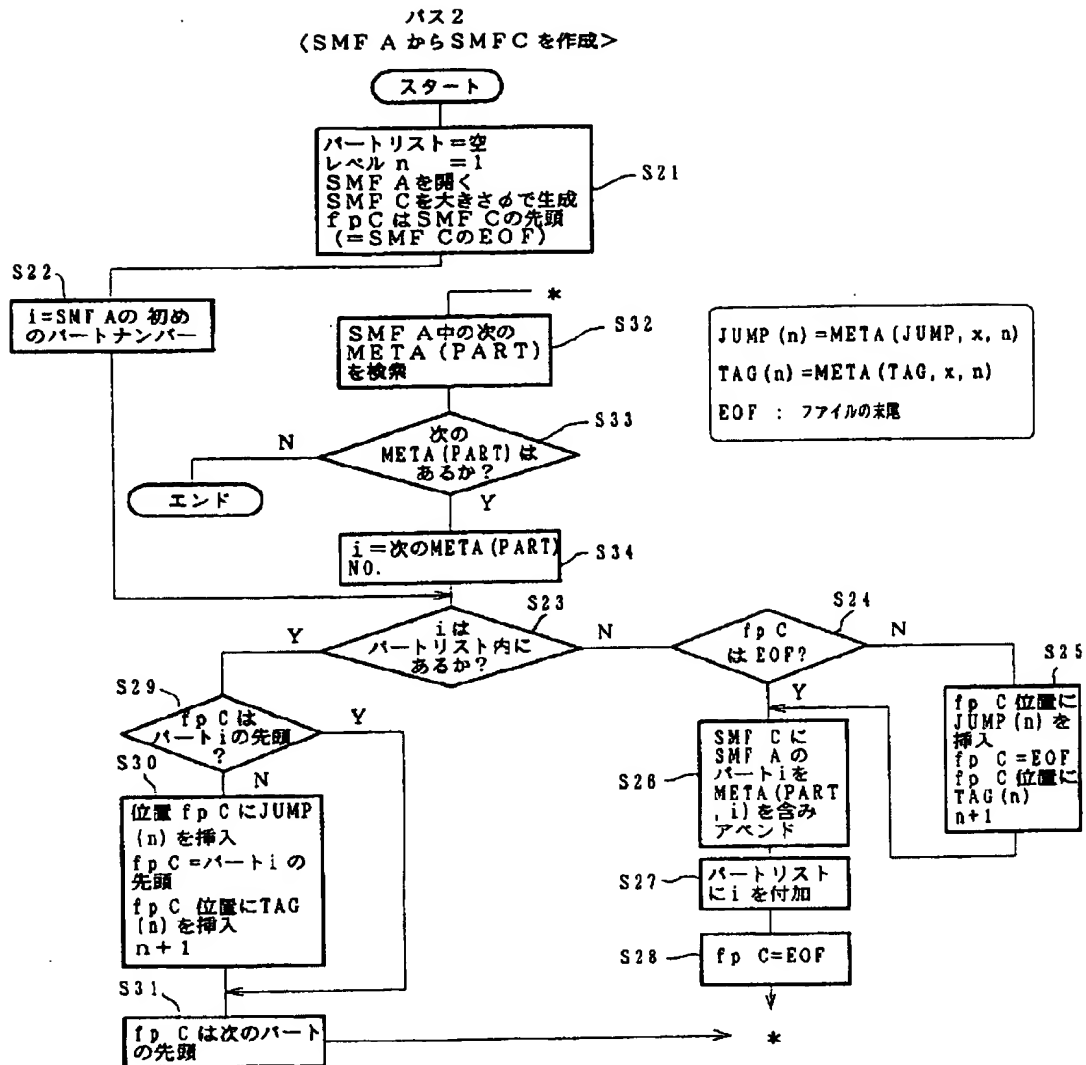
【図23】



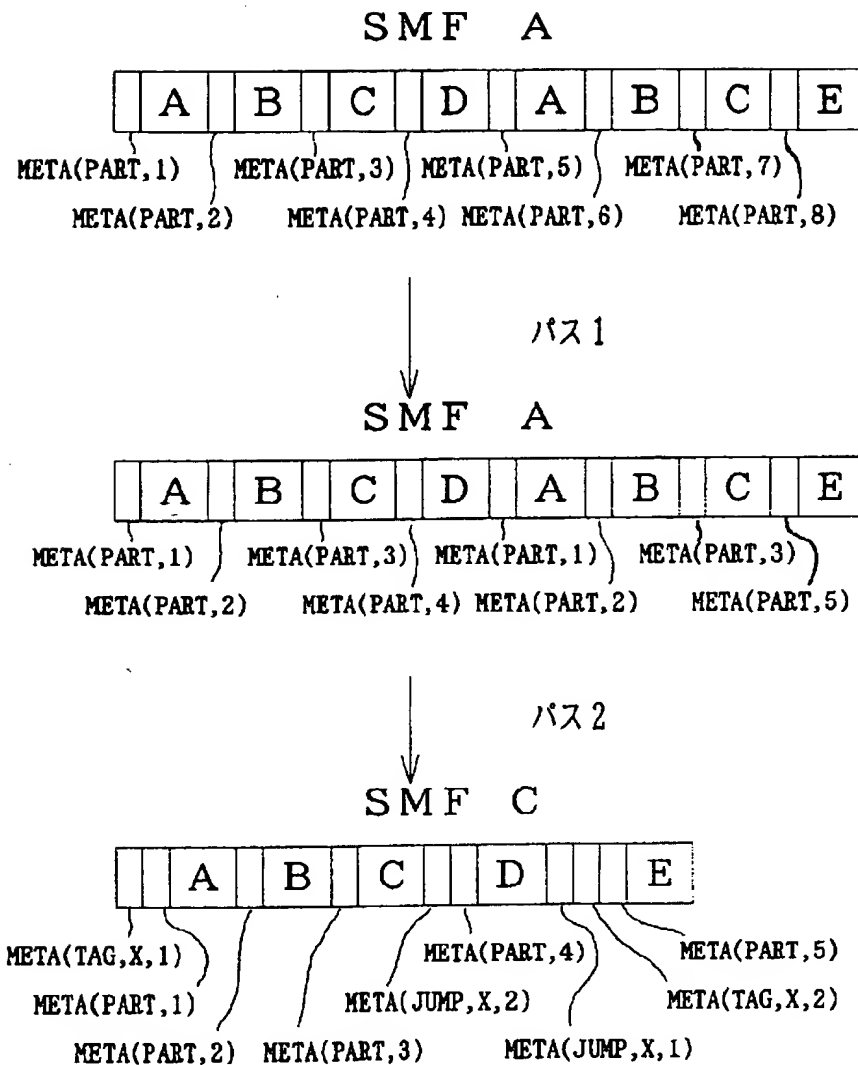
【図25】



【図 9】



【図 10】



【手続補正書】

【提出日】平成 6 年 1 2 月 1 5 日

【手続補正 1】

【補正対象書類名】明細書

【補正対象項目名】図 2 5

【補正方法】追加

【補正内容】

【図 2 5】D. S. (ダルセーニョ) を含んだ楽譜の例を示す図である。